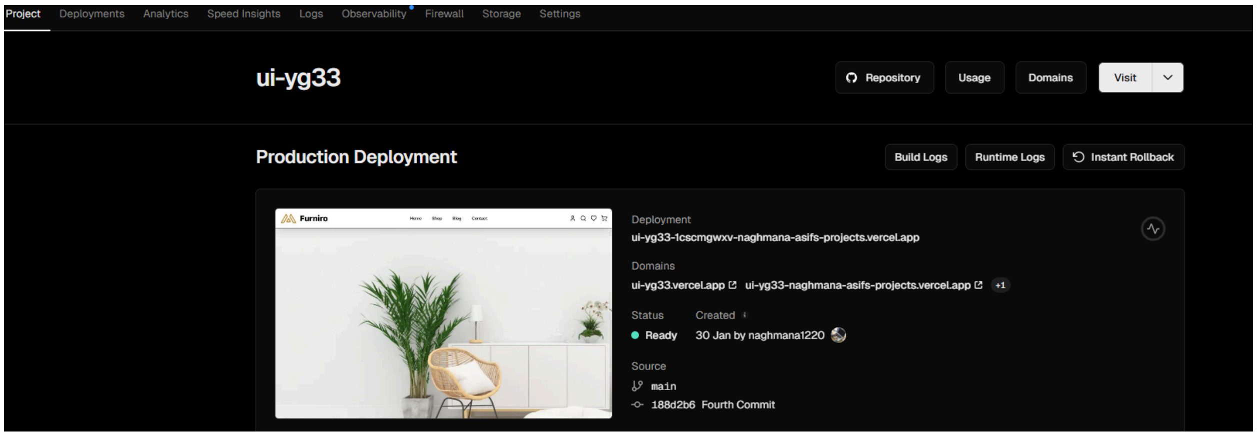
Day 6 - Deployment Preparation and Staging Environment Setup

On **Day 6**, we focus on preparing our marketplace for **deployment** by setting up a **staging environment**, configuring **hosting platforms**, and ensuring the application is ready for customers. Building on the **testing and optimization** work from **Day 5**, this stage ensures that the marketplace runs smoothly in a **production-like environment**. Additionally, we learn about **industry-standard practices** for managing different environments, including **non-production (TRN, DEV, SIT)** and **production (UAT, PROD, DR)**.

Step 1: Hosting Platform Setup

1) Choose Platform for deployment

For deploying our marketplace, we have chosen **Vercel** as the hosting platform. Vercel offers a **fast, reliable, and developer-friendly** deployment process, making it an ideal choice for **Next.js applications**. It provides seamless integration with **GitHub**, automatic deployments on every push, and built-in support for **environment variables**. Additionally, Vercel ensures **optimized performance, scalability, and security** for our staging and production environments.



ui-yg33

Production Deployment

Deployment
ui-yg33-1cscmgwxv-naghmana-asifs-projects.vercel.app

Domains
ui-yg33.vercel.app ui-yg33-naghmana-asifs-projects.vercel.app +1

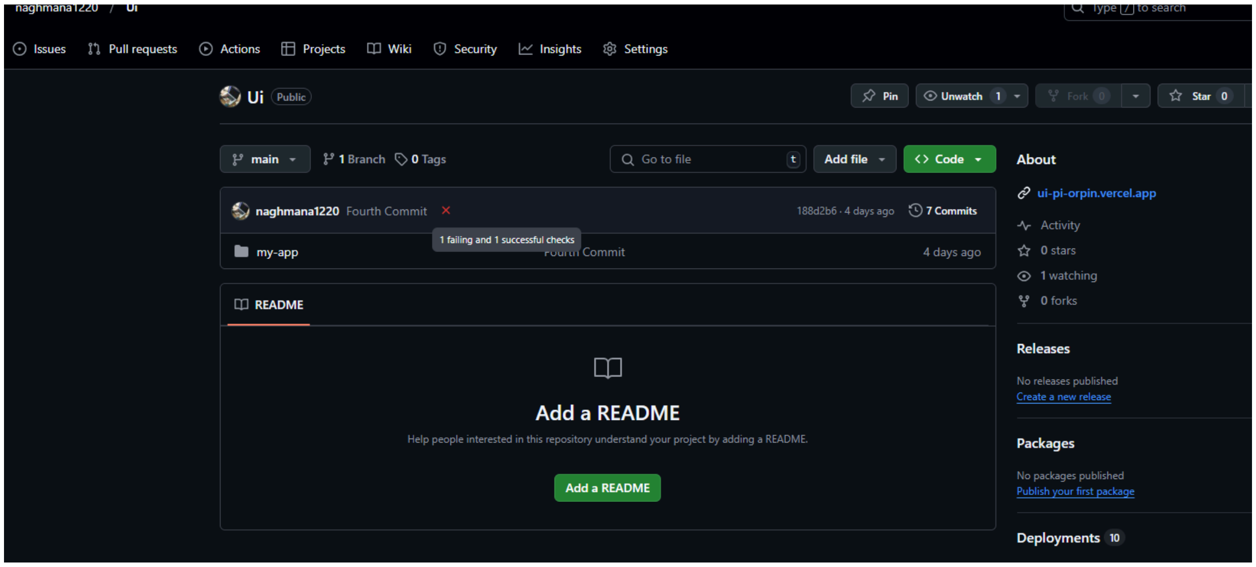
Status
Ready

Created
30 Jan by naghmana1220

Source
main
188d2b6 Fourth Commit

2) Connect Repository

I have successfully connected my **GitHub repository** to **Vercel** for seamless deployment. After logging into Vercel, I selected the **"New Project"** option and imported my repository



Step 2: Environment Variable Configuration

1)Create .env.local File

To securely manage sensitive information, I created a `.env.local` file in the root directory of my project. This file stores important **environment variables** such as API keys and database credentials.

For example:

NEXT_PUBLIC_SANITY_PROJECT_ID=your_project_id

NEXT_PUBLIC_SANITY_DATASET=production

API_KEY=your_api_key

2) Upload Variables to Hosting Platform

After creating the `.env.local` file, I securely uploaded the environment variables to **Vercel** to ensure they are available in the deployment environment. I navigated to **Vercel Dashboard** → **Project Settings** → **Environment Variables**, then added each variable manually. This ensures that sensitive data like **API keys** and **database credentials** remain secure and are not exposed in the codebase.

Step 3: Deploy to Staging

1. Deploy Application:

After configuring the environment variables, I deployed the application on **Vercel**. Vercel automatically detected my project settings and initiated the **build and deployment process**. Once the deployment was complete, I verified that the staging URL was working correctly, ensuring that the application loads without errors and functions as expected.

2) Validate Deployment:

Once the application was deployed on **Vercel**, I validated the deployment by thoroughly checking the staging environment. I ensured that the **build process** completed successfully without any errors. Then, I tested the application’s **basic functionality**, including key features like product listings, navigation, and cart functionality,and user interface to confirm that everything was working as expected.

Step 4: Staging Environment Testing

1. Testing Types:

To ensure the application's stability and performance, I conducted different types of testing in the **staging environment**:

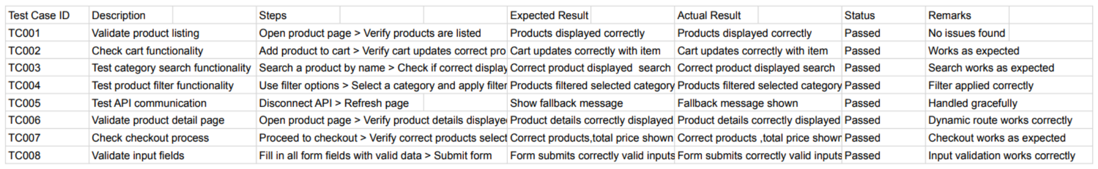
- **Functional Testing:** Verified that all features, such as product listing, search, and cart operations, work correctly.

- **Performance Testing:** Used tools like **Lighthouse** and **GTmetrix** to analyze page speed, responsiveness, and overall performance.

- **Security Testing:** Ensured secure API communications, validated input fields to prevent vulnerabilities like SQL injection, and checked for HTTPS enforcement.

2. Test Case Reporting:

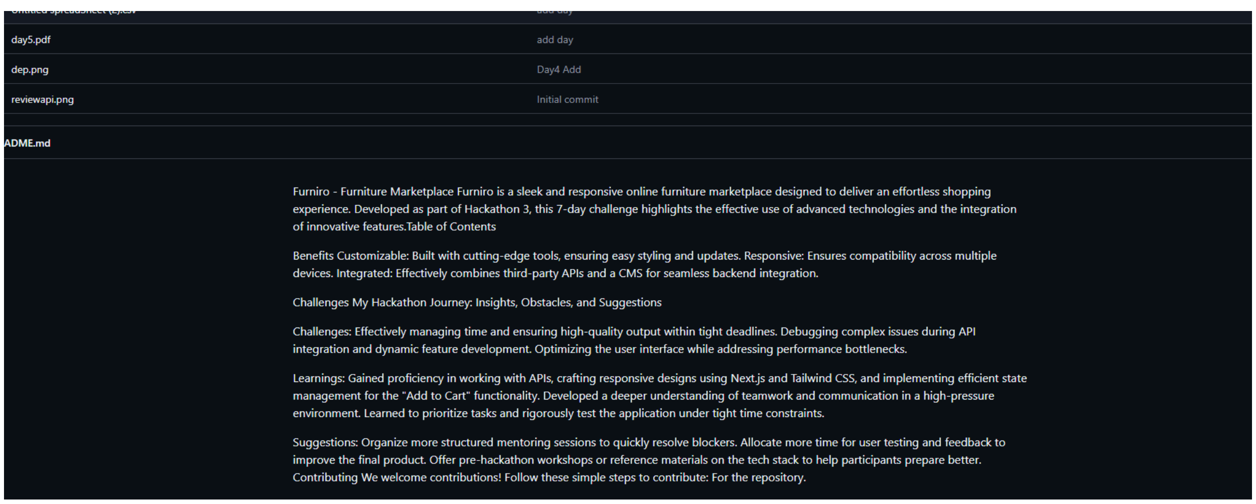
I documented all test cases and their outcomes in a **CSV file** for structured reporting. Each test case included details such as the **Test Case ID**, **Description**, **Steps**, **Expected Result**, **Actual Result**, **Status**, and **Remarks**. This helped track the progress of the testing phase and provided a clear record of any issues encountered and resolved



Test Case ID	Description	Steps	Expected Result	Actual Result	Status	Remarks
TC001	Validate product listing	Open product page > Verify products are listed	Products displayed correctly	Products displayed correctly	Passed	No issues found
TC002	Check cart functionality	Add product to cart > Verify cart updates correct pro	Cart updates correctly with item	Cart updates correctly with item	Passed	Works as expected
TC003	Test category search functionality	Search a product by name > Check if correct display	Correct product displayed search	Correct product displayed search	Passed	Search works as expected
TC004	Test product filter functionality	Use filter options > Select a category and apply filter	Products filtered selected category	Products filtered selected category	Passed	Filter applied correctly
TC005	Test API communication	Disconnect API > Refresh page	Show fallback message	Fallback message shown	Passed	Handled gracefully
TC006	Validate product detail page	Open product page > Verify product details displaye	Product details correctly displayed	Product details correctly displayed	Passed	Dynamic route works correctly
TC007	Check checkout process	Proceed to checkout > Verify correct products select	Correct products,total price shown	Correct products ,total price showr	Passed	Checkout works as expected
TC008	Validate input fields	Fill in all form fields with valid data > Submit form	Form submits correctly valid inputs	Form submits correctly valid inputs	Passed	Input validation works correctly

3. Performance Testing:

I have uploaded the performance testing report, generated using tools like Lighthouse, to my GitHub repository. This is done to ensure transparency and to share the performance analysis results.



day5.pdf	add day
dep.png	Day4 Add
reviewapi.png	Initial commit

ADME.md

Furniro - Furniture Marketplace Furniro is a sleek and responsive online furniture marketplace designed to deliver an effortless shopping experience. Developed as part of Hackathon 3, this 7-day challenge highlights the effective use of advanced technologies and the integration of innovative features.Table of Contents

Benefits Customizable: Built with cutting-edge tools, ensuring easy styling and updates. Responsive: Ensures compatibility across multiple devices. Integrated: Effectively combines third-party APIs and a CMS for seamless backend integration.

Challenges My Hackathon Journey: Insights, Obstacles, and Suggestions

Challenges: Effectively managing time and ensuring high-quality output within tight deadlines. Debugging complex issues during API integration and dynamic feature development. Optimizing the user interface while addressing performance bottlenecks.

Learnings: Gained proficiency in working with APIs, crafting responsive designs using Next.js and Tailwind CSS, and implementing efficient state management for the "Add to Cart" functionality. Developed a deeper understanding of teamwork and communication in a high-pressure environment. Learned to prioritize tasks and rigorously test the application under tight time constraints.

Suggestions: Organize more structured mentoring sessions to quickly resolve blockers. Allocate more time for user testing and feedback to improve the final product. Offer pre-hackathon workshops or reference materials on the tech stack to help participants prepare better. Contributing We welcome contributions! Follow these simple steps to contribute: For the repository.